ELSEVIER

Theoretical Computer Science 210 (1999) 261–275

Theoretical Computer Science

On the approximation of protein threading[1]

Tatsuya Akutsu*,[2], Satoru Miyano[2]

Human Genome Center, Institute of Medical Science, University of Tokyo, 4-6-1 Shirokanedai, Minato-ku, Tokyo 108 Japan

Abstract

In this paper, we study the protein threading problem, which was proposed for predicting a folded 3D protein structure from an amino acid sequence. Since this problem was already proved to be NP-hard, we study polynomial time approximation algorithms. We show several hardness results for the approximation, which includes a MAX SNP-hardness result. We also show approximation algorithms for a special case and a general case, where a graph representing interactions between amino acid residues is restricted to be planar in a special case. For this special case, we obtain a constant approximation ratio. © 1999—Elsevier Science B.V. All rights reserved

Keywords: Approximation algorithms; Protein threading; Protein folding; Computational biology

1. Introduction

The *protein folding* problem is, given an amino acid sequence (a string), to find its correctly folded 3D protein structure. It is one of the most important computational problems in molecular biology. Although this problem can be defined as a minimization problem, it is too hard to be solved directly.

Recently, an indirect approach was proposed [3, 5, 6, 8, 10, 13]. In this approach, given an amino acid sequence and a set of protein structures (structural patterns), a structure into which the sequence is most likely to fold is computed. To test whether or not a sequence is likely to fold into a structure, an *alignment* between spatial positions of a 3D structure and amino acids of a sequence is computed using a suitable *score function*. That is, an alignment which minimizes the total score (corresponding

* Corresponding author. E-mail: {takutsu,miyano}@ims.u-tokyo.ac.jp.

[1] A preliminary version of this paper has appeared in the proceedings of the first international conference on computational biology (RECOMB'97).

[2] Partially supported by the Grant-in-Aid for Scientific Research on Priority Areas, "Genome Science", of the Ministry of Education, Science, Sports and Culture in Japan.

0304-3975/99/$ – see front matter
PII: S0304-3975(98)00089-9

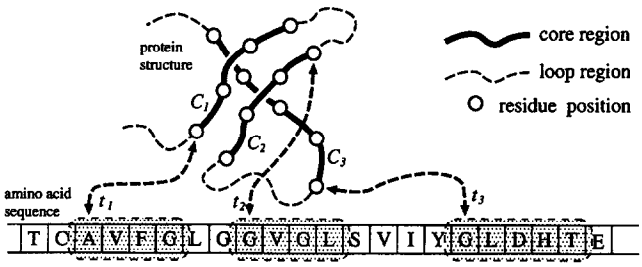

Fig. 1. The protein threading problem.

to potential energy) is computed. This minimization problem is called the *protein threading* problem, and an alignment between a sequence and a structure is called a *threading* (see Fig. 1) [5, 12, 13]. Note that, in Fig. 1, gaps (insertions and deletions of amino acids) are not allowed in *core regions*, but allowed only in *loop regions*, where a protein structure is partitioned into core regions and loop regions. This assumption was used in Refs. [5, 13] and seems biologically reasonable because insertions and deletions seldom occur in core regions. Thus, we also employ this assumption in this paper.

A variety of studies have been done for the protein threading problem [3, 5, 8, 10, 12, 13, 18]. However, there are only a few studies that try to find an *optimal threading* (i.e., a threading with the minimum score) [5, 12, 13, 18]. Bryant and Lawrence used exhaustive search to examine all possible threadings [5]. But, their method can only be applied to very small structures. Xu and Uberbacher proposed a polynomial time algorithm for a special case [18]. But, their algorithm does not seem to be applicable to many protein structures. Lathrop proved that the protein threading problem is NP-hard [12]. However, Lathrop and Smith applied the branch-and-bound search technique to the protein threading problem in a clever way and succeeded to compute optimal threadings for proteins of medium size [13].

Although Lathrop and Smith's algorithm is very nice, too long time is required if it is applied to a large protein structure. Thus, this paper studies a computational aspect of the protein threading problem. Since it was already proved to be NP-hard, we study (polynomial time) approximation algorithms. Of course, approximate solutions are different from optimal solutions which correspond to 3D shapes. However, they are useful from the following reasons: the correct structure could be selected if the guaranteed error bound were sufficiently small; approximate solutions might lead to optimal solutions if it is combined with local search; approximate solutions might be useful for speeding up the branch-and-bound procedure. Note that approximation algorithms have been already proposed for the protein folding problem (not the protein threading problem) [1, 9] although the considered models were too simple.

In this paper, we first show that the protein threading problem is MAX SNP-hard, from which a constant size lower bound of performance ratio follows under the assumption of $P \neq NP$ [2, 17]. Moreover, we show a result suggesting that approximation

of the problem is much harder, using an approximation preserving reduction from the DENSE-k-SUBGRAPH problem, for which only an $O(n^{0.3885})$ ratio approximation algorithm is known [11]. Next we consider a special case in which a graph representing interactions between cores (or, amino acid residues) is planar. This case corresponds to most of *β-sheet* substructures. Since the interactions between *α-helices* and other core regions are considered to be much weaker, it seems that this case covers most of protein structures. For this case, we show a polynomial time algorithm which approximates the optimal score within a constant factor. Finally, extending the idea used in a planar case, we obtain a polynomial time algorithm which approximates the optimal score within a factor of $O(\sqrt{M})$ where M denotes the number of pairs of cores having strong interactions.

In approximation algorithms, we use some decompositions of an edge set. These decompositions are similar to those in the *book embedding* [7]. However, our decompositions are different from those because the ordering of vertices is fixed in our case, whereas an arbitrary ordering can be selected in the book embedding.

2. The protein threading problem

As mentioned in Introduction, the protein threading problem (PROTEIN THREADING, in short) is, given a sequence and a 3D protein structure, to find an alignment (a threading) between the sequence and the structure with the minimum score. Lathrop and Smith defined this problem in a formal way [12, 13]. In this paper, we modify their definition into much simpler form without loss of generality as follows (see Fig. 1).

Input: Sequence $s = s_1 s_2 \ldots s_n$ over a fixed alphabet Σ (usually $|\Sigma| = 20$), core lengths $c_1, c_2, \ldots, c_m$, score function $g(i, j, t_i, t_j)$ $(g(i, j, t_i, t_j) \geqslant 0)$,

Output: m-tuple $\boldsymbol{t} = (t_1, \ldots, t_m)$ which maximizes $score(\boldsymbol{t}) = \sum_{i<j} g(i, j, t_i, t_j)$ under the condition that $1 \leqslant t_1$, $t_i + c_i \leqslant t_{i+1}$, $t_m + c_m \leqslant n + 1$.

Note that t_i means the first position of core C_i in sequence s, where C_i denotes the ith core (with length c_i). Thus, the condition of $1 \leqslant t_1$, $t_i + c_i \leqslant t_{i+1}$, $t_m + c_m \leqslant n + 1$ means that subsequences assigned to core regions must not overlap. Note also that, for each fixed pair of cores (C_i, C_j), $g(i, j, t_i, t_j)$ is a function from $\Sigma^{c_i} \times \Sigma^{c_j}$ to the set of positive reals, where t_i and t_j specify subsequences $s_{t_i} \ldots s_{t_i + c_i - 1}$ and $s_{t_j} \ldots s_{t_j + c_j - 1}$ respectively. Intuitively, $g(i, j, t_i, t_j)$ represents pseudo energy (or strength of interaction) between C_i and C_j when $s_{t_i} \ldots s_{t_i + c_i - 1}$ and $s_{t_j} \ldots s_{t_j + c_j - 1}$ are assigned to C_i and C_j respectively. Although various score functions have been proposed, most score functions can be expressed using this form and almost all score functions can be computed in polynomial time [13]. Thus, we assume that an arbitrary score function can be included as a part of input if its value is computable in polynomial time for all i, j, t_i, t_j.

The above definition may look strange since it is defined as a maximization problem, while the protein threading problem is usually defined as a minimization problem. However, the protein threading problem is intrinsically a maximization problem as

well as the protein folding problem is [1, 9]. Indeed, usual score functions can take negative values and the minimum score can become negative. Thus, by inverting the sign of values of score functions and by adding a constant factor, we can treat the protein threading problem as a maximization problem. This definition seems more natural when we consider approximation algorithms.

In Lathrop and Smith's definition, two kinds of score functions $g_1(i,t_i)$ and $g_2(i,j,t_i,t_j)$ are used, while only one kind is used in the above definition. However, letting $g(i,i+1,t_i,t_{i+1})=g_1(i,t_i)+g_2(i,i+1,t_i,t_{i+1})$, we can treat any score functions used by Lathrop and Smith. Although the lengths of loop regions are included in an input in Lathrop and Smith's definition, the effect of loop regions can be taken into account in the above definition by adding a length of a loop region to a length of a core region and by modifying $g(i,j,t_i,t_j)$ suitably. Therefore, in the above definition, there is no loss of generality.

We call $\boldsymbol{t}=(t_1,\ldots,t_m)$ a *threading* if $\boldsymbol{t}$ satisfies the condition in the definition of the problem (i.e., $1\leqslant t_1, t_i+c_i\leqslant t_{i+1}, t_m+c_m\leqslant n+1$). For an instance T of the problem, we associate a directed (multi) graph $G(V_T,E_T)$ such that $V_T=\{C_1,\ldots,C_m\}$, $E_T=\overline{E_T}\cup\{(C_i,C_{i+1})\mid 1\leqslant i<m\}$, where $\overline{E_T}=\{(C_i,C_j)\mid i<j \text{ and } \exists(t_i,t_j)(g(i,j,t_i,t_j)\neq 0)\}$, where similar graph structure is defined in Ref. [12]. Intuitively, $G(V_T,E_T)$ reflects the interactions between core regions: there exists an edge between two cores if their interaction is strong. $G(V_T,E_T)$ also reflects 3D structure of a protein because the interaction between two core regions is usually very weak if the distance between core regions is long. Note that $G(V_T,E_T)$ can have multi-edges, where the maximum multiplicity is at most 2.

In this paper, we consider two special cases: a case that the maximum vertex degree of $G(V_T,E_T)$ is bounded by a constant B, and a case that $G(V_T,E_T)$ is planar. The former case is called PROTEIN THREADING-B. Most protein structures correspond to this case because each core interacts with small number of other cores if we ignore weak interactions. Indeed, most scores become very small when the distance between residues exceeds a threshold value [12].

The planar case corresponds to β-sheet substructure, which appears in most core regions and is known as a kind of *secondary structure* (see Fig. 2) [4]. β-sheet consists of multiple β-strands. To classify β-sheet structures, topology diagram has been used [4]. In topology diagram, β-starnds are usually arranged parallel in a plane, and each β-strand strongly interacts with at most two neighbor β-strands. Thus, in most β-sheet

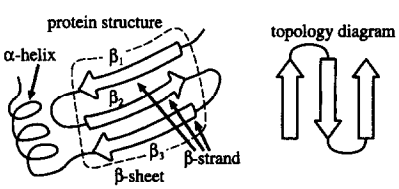

Fig. 2. β-sheet and topology diagram. In this case, β_2 strongly interacts with β_1 and β_3.

substructures, $G(V_T, E_T)$ can be considered a planar graph. Although other substructures (e.g., α-helices) can appear in core regions, the interactions between α-helices and other core regions are considered to be much weaker. Thus, good approximations would be obtained even if we do not consider the interactions between α-helices and other core regions. At least, this special case would be useful to identify β-strands in a given amino acid sequence. Identifying β-strands is also an important problem in molecular biology.

In this paper, we consider polynomial time approximation algorithms. Recall that the *performance ratio* (*approximation ratio*) of an approximation algorithm for a maximization problem is the worst-case ratio of the value of the optimal solution to the value of the approximate solution. If there exists an approximation algorithm with performance ratio $f(n)$ for a problem X where n denotes the size of an input, we say that X can be approximated within a factor of $f(n)$ [2].

3. Hardness results

First, we show that PROTEIN THREADING is MAX SNP-hard even if the maximum vertex degree is bounded by a constant B. Note that MAX SNP is a syntactically defined class [17] and it is proved that any MAX SNP-hard problem does not have a polynomial time approximation scheme unless P = NP [2]. That is, for each MAX SNP-hard problem, there exists a constant c (depending on a problem) such that this problem cannot be approximated within a factor of c in polynomial time unless P = NP. Note that the following theorem also gives a much simpler proof of NP-hardness of PROTEIN THREADING than that in Ref. [12].

Theorem 1. *PROTEIN THREADING-B is MAX SNP-hard.*

Proof. We use L-reduction from MAX CUT for graphs of bounded vertex degree $B-2$, where L-reduction is a standard reduction method for showing MAX SNP-hardness [17] and MAX CUT was already proved to be MAX SNP-complete even for graphs of maximum vertex degree 3 [17]. MAX CUT is, given an undirected graph $G(V,E)$, to find a subset $V' \subseteq V$ maximizing the cardinality of the cut (i.e., the number of edges between V' and $V - V'$). From $G(V,E)$ of bounded degree $B-2$ where $V = \{v_1, \ldots, v_n\}$, we construct an instance of PROTEIN THREADING-B in the following way:

$$s = \overbrace{01\,01\,\ldots\,01}^{n\text{-}01\text{'}s} \text{ over } \Sigma = \{0,1\}, \qquad c_1 = c_2 = \cdots = c_n = 1,$$

$$g(i,j,t_i,t_j) = \begin{cases} 1 & \text{if } i<j,\ (v_i,v_j) \in E \text{ and } s_{t_i} \neq s_{t_j}, \\ 0 & \text{otherwise.} \end{cases}$$

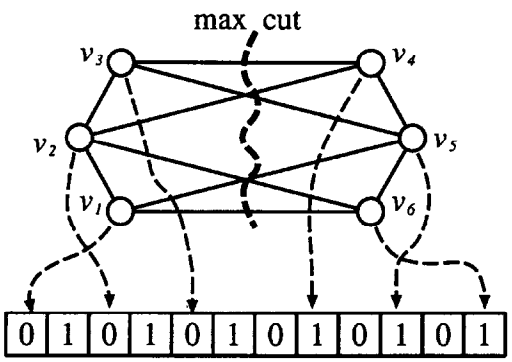

Fig. 3. Correspondence between the maximum cut and the optimal threading.

Then, each threading corresponds to a cut and the score of a threading is equal to the size of the corresponding cut (see Fig. 3).

Therefore, this reduction is L-reduction and the theorem follows. □

Note that, in the above proof (and the other proofs in this paper), we assume that an arbitrary (polynomial time computable) score function can be included as a part of input. Although this assumption is too strong, we may obtain the same or similar hardness results for limited class of score functions. For example, we can prove Theorem 1 even if $g(i,j,t_i,t_j)$ depends only on subsequences assigned to core regions (i.e., score function is a function from $\sum^* \times \sum^*$ to the set of real numbers) by associating different substrings with different vertices using binary codes. However, considering such a score function is not adequate because it does not reflect geometric or chemical structure of a protein. Since simpler proofs are better and it is not known which subclass of score functions is adequate, we assume that an arbitrary score function can be included as a part of input.

Next, we show a result suggesting that approximation of PROTEIN THREADING is much harder, using a reduction from the DENSE-k-SUBGRAPH problem [11]. DENSE-k-SUBGRAPH is, given an undirected graph $G(V,E)$ and an integer k, to find a subset $V' \subseteq V$ of cardinality k with maximum number of edges (in the induced subgraph by V'). Although no lower bound of performance ratio has been proved, DENSE-k-SUBGRAPH is considered to be hard to approximate. Currently, an approximation algorithm by Kortsarz and Peleg achieves the best performance ratio of $O(n^{0.3885})$ [11].

Theorem 2. *If PROTEIN THREADING can be approximated within a factor of $f(|s|)$ in polynomial time, DENSE-k-SUBGRAPH can be approximated within a factor of $f(|V|^2)$ in polynomial time.*

Proof. From an instance $(G(V,E),k)$ of DENSE-k-SUBGRAPH where $V=\{v_1,\ldots,v_n\}$, we construct an instance of PROTEIN THREADING in the following way:

$$s=\overbrace{00\ldots0}^{n-1}a_1\overbrace{00\ldots0}^{n-1}a_2\overbrace{00\ldots0}^{n-1}\ldots\overbrace{00\ldots0}^{n-1}a_k\overbrace{00\ldots0}^{n^2-kn} \text{ over } \Sigma=\{0,1\}$$

where $a_i=1$ for all i, $c_1=c_2=\cdots=c_n=1$,

$$g(i,j,t_i,t_j)=\begin{cases}1 & \text{if } i<j,\ (v_i,v_j)\in E \text{ and } s_{t_i}=s_{t_j}=1,\\ 0 & \text{otherwise.}\end{cases}$$

Then, a subset of cores $\{C_i \mid s_{t_i}=1\}$ corresponds to a subset V'. From this observation, it is easy to see that the maximum score in PROTEIN THREADING is equal to the maximum number of edges in DENSE-k-SUBGRAPH. Moreover, given a threading $\boldsymbol{t}$, we can obtain $V'\subseteq V$ of cardinality at most k having $score(\boldsymbol{t})$ edges in polynomial time. Since $|s|$ (length of s) is n^2, the theorem holds. □

Note that in the above theorem, a factor of $f(|V|^2)$ can be improved to a factor of $f((k+1)|V|)$ by replacing n^2-kn with n in the construction of s.

If the maximum degree of $G(V,E)$ is bounded by a constant, there is a trivial constant ratio approximation algorithm for DENSE-k-SUBGRAPH. In the construction in Theorem 2, an instance of PROTEIN THREADING-B is constructed from such a graph. Thus, the above theorem does not suggest the hardness of PROTEIN THREADING-B. However, we can show a theorem that suggests the hardness of PROTEIN THREADING-B.

Theorem 3. *If PROTEIN THREADING-B can be approximated within a factor of $f(|s|)$ in polynomial time, DENSE-k-SUBGRAPH can be approximated within a factor of $f(2|V|^3)$ in polynomial time.*

Proof. Modify the construction in Theorem 2 as follows:

$$s=\overbrace{00\ldots0}^{2n^2-n}b_1\overbrace{00\ldots0}^{2n^2-n}b_2\overbrace{00\ldots0}^{2n^2-n}\ldots\overbrace{00\ldots0}^{2n^2-n}b_k\overbrace{00\ldots0}^{2n^3-2kn^2} \text{ over } \Sigma=\{0,1\}$$

where each b_i is a sequence of 1's with length n,

$c_1=c_2=\cdots=c_{2n^2}=1$,

$g(2n(i-1)+j,2n(j-1)+i,t_{2n(i-1)+j},t_{2n(j-1)+i})=1$ if $0<i<j\leqslant n$,

$(v_i,v_j)\in E$ and $s_{t_{2n(i-1)+j}}=s_{t_{2n(j-1)+i}}=1$, otherwise $g(i,j,t_i,t_j)=0$.

In this case, $G(V_T,E_T)$ is a graph of bounded degree 3. Note that consecutive cores $C_{2n(i-1)+1}C_{2n(i-1)+2}\ldots C_{2n(i-1)+n}$ correspond to a vertex v_i, and cores $C_{2n(i-1)+n+1}$ $C_{2n(i-1)+n+2}\ldots C_{2n(i-1)+2n}$ are introduced so that one b_j does not correspond to multiple

vertices. Thus, the theorem can be proved using a similar argument as in Theorem 2. □

Note that, as in Theorem 2, a factor of $f(2|V^3|)$ can be improved to $f(2(k+1)|V^2|)$ by replacing $2n^3 - 2kn^2$ with $2n^2$.

Next, we consider a case that $G(V_T, E_T)$ is planar. For this case, we will show a constant ratio approximation algorithm in the next section. But, even in this case, the problem remains NP-hard as shown below.

Theorem 4. *PROTEIN THREADING-B is* NP*-hard even if* $G(V_T, E_T)$ *is planar.*

Proof. We use a reduction from the longest common subsequence problem (LCS) over a binary alphabet, which was already shown to be NP-hard [15]. LCS is, given strings $s^1 = s^1_1 \ldots s^1_{n_1}$, $s^2 = s^2_1 \ldots s^2_{n_2}, \ldots, s^k = s^k_1 \ldots s^k_{n_k}$ over $\Sigma' = \{0,1\}$ and an integer L, to decide whether or not there exists a string s' of length L such that s' is a subsequence of s^i for all $i \leqslant k$. We assume without loss of generality that k is even. Let $(s^i)^{-1}$ denote the reversed string of s^i (i.e., $(s^i)^{-1} = s^i_{n_i} s^i_{n_i-1} \ldots s^i_1$).

From an instance of LCS, we construct an instance of PROTEIN THREADING as follows:

$$s = b\, s^1\, aa\, (s^2)^{-1}\, aa\, s^3\, aa\, (s^4)^{-1}\, aa \ldots s^{k-1}\, aa\, (s^k)^{-1}\, b \text{ over } \Sigma = \{0,1,a,b\},$$

$$c_1 = c_2 = \cdots = c_{k(L+2)} = 1,$$

$$g(i,j,t_i,t_j) = \begin{cases} 1 & \text{if } (1 \leqslant \exists p \leqslant k-1)(1 \leqslant \exists q \leqslant L)(i = (L+2)p - q \\ & \text{and } j = (L+2)p + 1 + q \text{ and } s_{t_i} = s_{t_j} \in \{0,1\}), \\ \alpha & \text{if } (1 \leqslant \exists p \leqslant k-1)(i = (L+2)p \text{ and } j = (L+2)p + 1 \\ & \text{and } s_{t_i} = s_{t_j} = a), \\ 0 & \text{otherwise.} \end{cases}$$

Note that $G(V_T, E_T)$ becomes a planar graph in this case (see Fig. 4). We let $\alpha \gg kL$. Then, it is easy to see that there exists a common subsequence s' of length L if and only if there exists a threading $\boldsymbol{t}$ such that $score(\boldsymbol{t}) \geqslant (k-1)(L+\alpha)$. □

4. Approximation in a planar case

In this section, we show a constant ratio approximation algorithm for a special case of PROTEIN THREADING that an associated graph $G(V_T, E_T)$ is planar.

To develop an approximation algorithm, we partition a set of edges $\overline{E_T}$ into three subsets (see Fig. 5): E_u (a set of *upper edges*), E_ℓ (a set of *lower edges*), E_o (a set of *loop edges*). First we show that an optimal threading can be obtained using a simple dynamic programming algorithm if $E_\ell = E_o = \emptyset$ or $E_u = E_o = \emptyset$. Next we show an approximation algorithm with performance ratio 2 for a case of $E_\ell = E_u = \emptyset$. Combining those, we obtain an approximation algorithm with performance ratio 4.

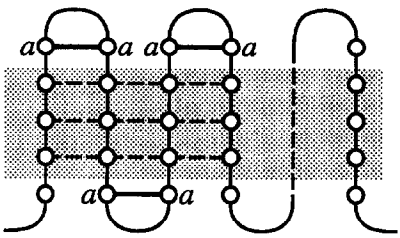

Fig. 4. $G(V_T, E_T)$ appeared in the proof of Theorem 4. Shaded part corresponds to LCS.

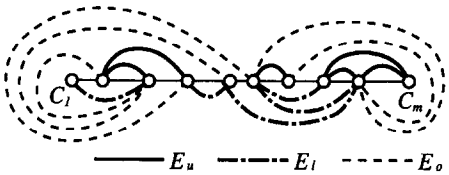

Fig. 5. Partition of $\overline{E_T}$ into E_u, E_ℓ and E_o. In this figure, cores are arranged on a horizontal line where each core is denoted by a circle.

Although we only describe a method to compute scores of approximate threadings, it can be modified for computing such threadings.

4.1. An algorithm for upper edges

In this subsection, we assume that all edges in $\overline{E_T}$ are upper edges (i.e., $E_\ell = E_o = \emptyset$). Note that, in this case, $G(V_T, E_T \cup \{(C_m, C_1)\})$ becomes an outplanar graph. Obviously, a case of $E_u = E_o = \emptyset$ can be treated in an analogous way. First, we can see the following property from the fact that any two upper edges do not cross.

Observation 5. *Assume that $(C_i, C_j) \in E_u$ holds and t_i, t_j are fixed. Then, the positions of $t_{i+1}, t_{i+2}, \ldots, t_{j-1}$ do not affect the scores of edges whose endpoints do not include any of $C_{i+1}, \ldots, C_{j-1}$.*

From this observation, we can develop a simple dynamic programming algorithm in the following way.

We define score $w(i, j, x, y)$ by

$$w(i,j,x,y) = \max_{t} \left\{ \sum_{i \leq h < k \leq j} g(h,k,t_h,t_k) \right\},$$

where the maximum is taken from all threadings t such that $t_i = x$ and $t_j = y$ $(x < y)$. That is, $w(i, j, x, y)$ denotes the maximum score for cores $C_i, C_{i+1}, \ldots, C_j$ under the condition that positions of C_i and C_j are fixed to x and y respectively. For each pair (C_i, C_j) such that $(C_i, C_j) \in E_u$ or $j = i + 1$, we compute $w(i, j, x, y)$ (for all x, y).

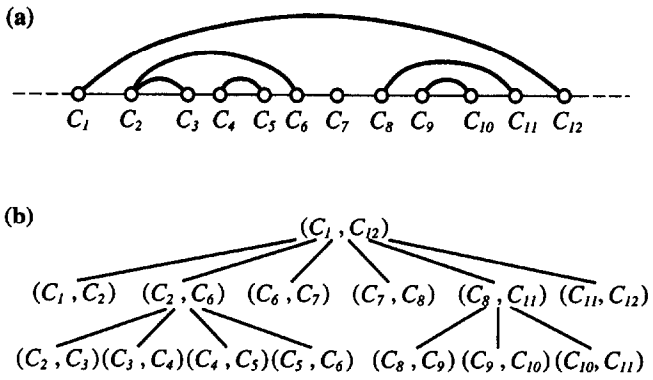

Fig. 6. Example of a tree structure associated with a set of upper edges.

For each pair (C_i, C_j) such that $(C_i, C_j) \in E_u$ or $j = i+1$, $(C_{i'}, C_{j'}) \in E_u$ is called an *ancestor* if either $i' \leqslant i < j < j'$ or $i' < i < j \leqslant j'$ holds. Moreover, $(C_{i'}, C_{j'})$ is called a *parent* of (C_i, C_j) if there exists no $(C_{i''}, C_{j''})$ such that $(C_{i''}, C_{j''})$ is an ancestor of (C_i, C_j) and $(C_{i'}, C_{j'})$ is an ancestor of $(C_{i''}, C_{j''})$. Then, a tree structure is induced by this relationship (see Fig. 6). Following this tree structure, we compute $w(i,j,x,y)$'s from leaves to the root.

For each (C_i, C_j) corresponding to a leaf (i.e., $j = i+1$), we let

$$w(i,j,x,y) = g(i,j,x,y).$$

For each (C_i, C_j) corresponding to an internal node, let $(C_i = C_{r_1}, C_{r_2}, \ldots, C_{r_p} = C_j)$ be a sequence of cores such that (C_i, C_j) is a parent of $(C_{r_q}, C_{r_{q+1}})$, where $r_1 < r_2 < \cdots < r_p$. For example, $(C_1, C_2, C_6, C_7, C_8, C_{11}, C_{12})$ is such a sequence for (C_1, C_{12}) in Fig. 6(a). Here, we can assume that values of $w(r_q, r_{q+1}, x, y)$'s are already computed before computing values of $w(i,j,x,y)$'s. Then, we let

$$w(i,j,x,y) = g(i,j,x,y) + w_1(j,x,y),$$

where $w_1(j,x,y)$ is computed by the following dynamic programming procedure:

$$w_1(r_1,x,z) = \begin{cases} 0 & \text{if } z = x, \\ -\infty & \text{otherwise,} \end{cases}$$

$$w_1(r_{k+1},x,z) = \max_{u \leqslant z - c_{r_k}} \{w_1(r_k,x,u) + w(r_k, r_{k+1}, u, z)\}.$$

Note that $w_1(r_k,x,z)$ denotes the maximum score for cores $C_{r_1}, C_{r_1+1}, \ldots, C_{r_k}$ under the condition that positions of C_{r_1} and C_{r_k} are fixed to x and z respectively.

Since it is easy to see that the score of an optimal threading is given by $\max_{x,y} w(1,m,x,y)$ where we assume without loss of generality that $(C_1, C_m) \in E_u$, we analyze the time complexity. First note that there are $O(m)$ edges in E_T because $G(V_T, E_T)$ is planar (with multiplicity at most 2) and $|V_T| = m$. For all (C_i, C_j) such that $(C_i, C_j) \in E_u$ or

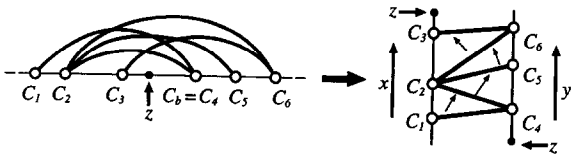

Fig. 7. Example of a very simple loop case, where loop edges are described in upper half plane. In this case, an optimal threading can be computed using dynamic programming.

$j=i+1$ and for all x, y, we must compute $w(i,j,x,y)$. Therefore, the number of $w(i,j,x,y)$'s to be computed is $O(mn^2)$. Next note that, for each $w(i,j,x,y)$, we must compute $O(pn)$ values of $w_1(r_k,x,z)$ (recall that $r_1=i$ and $r_p=j$). Since $O(n)$ time is required per $w_1(r_k,x,z)$, $O(pn^2)$ time is required per $w(i,j,x,y)$. Since $\sum_{(C_i,C_j)\in E_u} p$ is $O(m)$, the total computation time is $O(mn^4)$.

However, w_1's used to compute $w(i,j,x,y)$ can also be used to compute $w(i,j,x,y')$ for $y'>y$. Using this fact, the time complexity can be reduced to $O(mn^3)$.

Lemma 6. *An optimal threading can be computed in* $O(mn^3)$ *time if* $E_\ell=E_o=\emptyset$ *or* $E_u=E_o=\emptyset$, *where we ignore the time for computing values of a score function.*

Note that from the construction in Theorem 4, computing an optimal threading is NP-hard if $E_\ell\neq\emptyset$ and $E_u\neq\emptyset$.

4.2. An approximation algorithm for loop edges

In this subsection, we assume that all edges in $\overline{E_T}$ are loop edges (i.e., $E_u=E_\ell=\emptyset$) until Theorem 9. Such a case is called a *loop* case. Note that, in a loop case, the following property holds: for any two edges $(C_i,C_j),(C_{i'},C_{j'})\in E_o$, $j\leqslant j'$ if $i\leqslant i'$.

First we begin with a very simple case (see Fig. 7) in which the following condition holds: there exists a number b $(1<b\leqslant m)$ such that every edge $(C_i,C_j)\in E_o$ must satisfy $i<b\leqslant j$. For each i,j such that $1\leqslant i<b$ and $b\leqslant j\leqslant m$, let $w_b(i,j,x,y,z)$ be the maximum score $(\max_t \sum_{1\leqslant h\leqslant i, b\leqslant k\leqslant j} g(h,k,t_h,t_k))$ under the condition that $t_h+c_h\leqslant z$ and $t_k\geqslant z$. Then $w_b(i,j,x,y,z)$ is determined by the following recurrence (#) (see Fig. 7):

$$w_b(1,b,x,y,z)=\begin{cases} g(1,b,x,y) & \text{if } x+c_1\leqslant z \text{ and } y\geqslant z,\\ -\infty & \text{otherwise,}\end{cases}$$

$$w_b(i,j,x,y,z)=g(i,j,x,y)+\begin{cases}\max_{y'\leqslant y-c_{j-1}} w_b(i,j-1,x,y',z) & \text{if } (C_i,C_{j-1})\in E_o,\\ \max_{x'\leqslant x-c_{i-1}} w_b(i-1,j,x',y,z) & \text{otherwise.}\end{cases}$$

By computing $\max_{x,z,y} w_b(1,m,x,y,z)$ where $x<z\leqslant y$, we can obtain the maximum score in this very simple case. Since the total number of $w_b(i,j,x,y,z)$'s is $O(mn^3)$ and

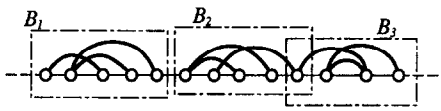

Fig. 8. Example of a simple loop case. E_0 can be partitioned into blocks.

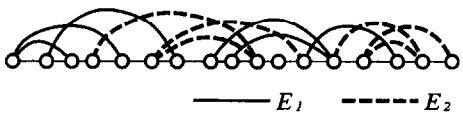

Fig. 9. Partition of E_0 into E_1 and E_2.

$O(n)$ time is required per $w_b(i,j,x,y,z)$, the total computation time is $O(mn^4)$. However, $\max_{y' \leqslant y+1-c_{j-1}} w_b(i,j-1,x,y',z)$ can be computed from $\max_{y' \leqslant y-c_{j-1}} w_b(i,j-1, x,y',z)$ in $O(1)$ time. Therefore, $O(1)$ time is required per $w_b(i,j,x,y,z)$ and thus the total computation time is $O(mn^3)$.

Next we consider a case that there exists no three edges (C_i, C_j), $(C_{i'}, C_{j'})$, $(C_{i''}, C_{j''})$ in E_0 such that $i', i'', j' \leqslant i$ and $i < j''$ (see Fig. 8). In this case, E_0 can be partitioned into disjoint sets (blocks) $B_1, B_2, \ldots, B_h$ where each block B_i corresponds to a very simple case (see Fig. 8). This case is called a *simple loop case*, and we say that E_0 is *simple*. Let $l(B_k) = \min\{i \mid (C_i, C_j) \in B_k\}$, $m(B_k) = \min\{j \mid (C_i, C_j) \in B_k\}$, and $r(B_k) = \max\{j \mid (C_i, C_j) \in B_k\}$. For convenience, we define $l(B_{h+1}) = r(B_h)$ and $m(B_{h+1}) = r(B_{h+1}) = m$.

Lemma 7. *An optimal threading can be computed in $O(mn^3)$ time if E_0 is simple, where we ignore the time for computing values of a score function.*

Proof. Partition E_0 into $B_1, B_2, \ldots, B_h$. For any B_i and for any x, y $(1 \leqslant x \leqslant y \leqslant n)$, an optimal score under the condition that $t_{l(B_i)} = x$ and $t_{r(B_i)} = y$ can be computed using the recurrence (#). Since $r(B_i) \leqslant l(B_{i+1})$ holds, an optimal score for E_0 can be computed using a simple dynamic programming algorithm. Since the score for each B_i is computed in $O(|B_i| n^3)$ time and this simple dynamic programming procedure takes $O(mn^3)$ time, the total computation time is $O(mn^3)$. □

Next we show that a loop case problem can be reduced to two simple loop case problems (see Fig. 9).

Lemma 8. *E_0 can be partitioned in $O(m^2)$ time into two disjoint sets E_1 and E_2 each of which corresponds to a simple loop case.*

Proof. We use the following simple greedy algorithm:

begin
$E_1 := \emptyset$;

```
  for i := 1 to m − 1 do
    for j := i + 1 to m do
      if (C_i, C_j) ∈ E_o and E_1 ∪ {(C_i, C_j)} is simple
      then E_1 := E_1 ∪ {(C_i, C_j)};
  E_2 := E_o − E_1
end
```

Obviously, E_1 obtained by this algorithm is simple. Here, we consider a partition of E_1 into $B_1, B_2, \ldots, B_h$. Then, each edge $(C_i, C_j) \in E_2$ satisfies the following condition: there exists k such that $m(B_k) \leqslant i \leqslant r(B_k)$ and $r(B_k) \leqslant j \leqslant m(B_{k+1})$. Therefore, E_2 also becomes simple. Moreover, this algorithm can be implemented so that it works in $O(m^2)$ time by maintaining $l(B_i)$, $m(B_i)$ and $r(B_i)$ appropriately. □

Combining Lemmas 7 and 8, we can see that PROTEIN THREADING can be approximated within a factor 2 in $O(mn^3)$ time if $E_u = E_\ell = \emptyset$.

Finally, we obtain the following theorem.

Theorem 9. *For any polynomial time computable score function, PROTEIN THREADING can be approximated within a factor* 4 *in polynomial time if* $G(V_T, E_T)$ *is planar.*

Proof. We use the following simple algorithm. First compute a planar embedding of $G(V_T, E_T)$ and a partition of $\overline{E_T}$ into E_u, E_ℓ, E_1, E_2. Next compute an optimal threading t_u by letting $E_\ell = E_1 = E_2 = \emptyset$. t_ℓ, t_1, t_2 are computed in a similar way. Finally select the one ($t_{\max}$) having the maximum score from t_u, t_ℓ, t_1, t_2.

It is easy to see that $score(t_{\max}) \geqslant (1/4) score(t_{\text{opt}})$ holds since

$$score(t_{\text{opt}}) \leqslant score(t_u) + score(t_\ell) + score(t_1) + score(t_2),$$

where t_{opt} denotes an optimal threading. Therefore, the performance ratio of this simple algorithm is at least 4.

Since a planar embedding of $G(V_T, E_T)$ can be computed in $O(|V_T|) = O(m)$ time [16], we can obtain a partition of $\overline{E_T}$ into E_u, E_ℓ, E_1, E_2 in $O(m^2)$ time. Therefore, the total computation time is $O(mn^3)$ from Lemma 6, Lemma 7 and $m \leqslant n$, except the time for computing values of a score function.

Since we need values of $g(i, j, t_i, t_j)$ for $O(mn^2)$ quadruplets, the total computation time is bounded by some polynomial. □

5. Approximation in a general case

In a planar case, an input edge set is decomposed into four edge sets, for each of which an optimal threading can be computed in polynomial time. This idea can be generalized for a general case.

In a general case, we partition $\overline{E_T}$ into $E_1, E_2, \ldots, E_N$ as follows. We sort edges of $\overline{E_T}$ so that (C_i, C_j) precedes to (C_h, C_k) if and only if $i<h$ holds or $i=h$ and $j<k$ hold. Then, we identify $I=\{j \mid (C_i, C_j) \in \overline{E_T}\}$ with a sequence of integers.

It is well known that for any sequence of integers $(i_1, i_2, \ldots, i_{k^2+1})$, there exists a monotonically increasing or decreasing subsequence with length at least k [14]. From sequence I, we find a longest monotone subsequence I' and we let E_1 be the set of edges corresponding to I'. From $\overline{E_T} - E_1$, we find a longest monotone subsequence and we obtain E_2 in the same way. We repeat this procedure until the remaining edge set becomes empty.

Note that each E_i becomes either a set of upper edges (in a case of decreasing subsequence) or a set of loop edges (in a case of increasing subsequence). If E_i is a set of loop edges, we decompose E_i into E_i' and E_i'' as in Lemma 8. For each of decomposed edge sets, we compute an optimal threading. Finally, as in Theorem 9, we select a threading with the maximum score as an approximate solution.

Using this algorithm, we have the following theorem.

Theorem 10. *For any polynomial time computable score function, PROTEIN THREADING can be approximated within a factor of* $O(\sqrt{M})$ *in polynomial time, where* $M=|\overline{E_T}|$.

Proof. First we evaluate the number of edge sets obtained by the decomposition. Since the number of edge sets is at most $2N$, it is sufficient to bound N. Let k_i be the number of remaining edges after $E_1, \ldots, E_i$ are removed. We assume without loss of generality that M is a power of 2. We prove that $k_{\lceil\sqrt{M}\rceil} < M/2$ holds for sufficiently large M.

Suppose $k_{\lceil\sqrt{M}\rceil} < M/2$ does not hold. Then, from the property on the monotone subsequence, the size of E_i is greater than $\lceil\sqrt{M/4}\rceil$ for each $i \leqslant \lceil\sqrt{M}\rceil$. Thus, we have $k_{\lceil\sqrt{M}\rceil} \leqslant M - \lceil\sqrt{M}\rceil\lceil\sqrt{M/4}\rceil < M/2$, which is a contradiction.

Using similar arguments repeatedly, we can prove $k_{f(i)} < M/2^i$ where $f(i) = \sum_{j=1}^{i} \lceil\sqrt{M/2^{j-1}}\rceil$. Therefore, the number of edge sets N is $O(\sqrt{M})$ and thus the approximation ratio is $O(\sqrt{M})$.

Next we analyze the time complexity. Finding a longest monotone subsequence of I can be done in $O(|I|^2)$ time using a simple dynamic programming procedure. Therefore, the time required for computing a decomposition is $\sum_{i=0}^{\lceil\log M\rceil} O((M/2^i)^2\sqrt{M}) = O(M^{2.5})$. For each E_i, an optimal threading can be computed in $O(mn^3)$ time. Therefore, the total computation time is $O(M^{2.5} + mn^3\sqrt{M})$ except the time for computing values of a score function. Since $M \leqslant m^2$, it is $O(m^5 + m^2n^3)$.

Since we need values of $g(i, j, t_i, t_j)$ for $O(Mn^2)$ quadruplets, the total computation time is bounded by some polynomial. □

Note that the above approximation ratio is $O(\sqrt{m})$ if the maximum vertex degree of $G(V_T, E_T)$ is bounded by a constant.

Acknowledgements

The authors would like to thank anonymous referees for helpful comments.

References

[1] R. Agarwala, S. Batzoglou, V. Dančík, S.E. Decatur, M. Farach, S. Hannenhalli, S. Skiena, Local rules for protein folding on a triangular lattice and generalized hydrophobicity in the HP model, in: Proc. 8th ACM-SIAM Symp. Discrete Algorithms, 1997, pp. 390–399.

[2] S. Arora, C. Lund, R. Motwani, M. Sudan, M. Szegedy, Proof verification and hardness of approximation algorithms, in: Proc. 33rd IEEE Symp. Foundations of Computer Science, 1992, pp. 14–23.

[3] J.U. Bowie, R. Lüthy, D. Eisenberg, A method to identify protein sequences that fold into a known three-dimensional structures, Science 253 (1991) 164–170.

[4] C. Branden, J. Tooze, Introduction to Protein Structure, Garland Publishing, New York, 1991.

[5] S.H. Bryant, C.E. Lawrence, An empirical energy function for threading protein sequence through the folding motif, PROTEINS: Structure, Function, and Genetics 16 (1993) 92–112.

[6] C. Chothia, One thousand families for the molecular biologist, Nature 357 (1992) 543–544.

[7] F.R.K. Chung, F.T. Leighton, A.L. Rosenberg, Embedding graphs in books: a layout problem with applications to VLSI design, SIAM J. Discrete Math. 8 (1987) 33–58.

[8] A. Godzik, J. Skolnick, Sequence-structure matching in globular proteins: application to supersecondary and tertiary structure determination, Proc. National Academy of Science USA 89 (1992) 12098–12102.

[9] W.E. Hart, S.C. Istrail, Fast protein folding in the hydrophobic-hydrophilic model within three-eights of optimal, J. Comput. Biol. 3 (1996) 53–96.

[10] D.T. Jones, W.R. Taylor, J.M. Thornton, A new approach to protein fold recognition, Nature 358 (1992) 86–89.

[11] G. Kortsarz, D. Peleg, On choosing a dense subgraph, in: Proc. 34th IEEE Symp. Foundations of Computer Science, 1993, pp. 692–701.

[12] R.H. Lathrop, The protein threading problem with sequence amino acid interaction preferences is NP-complete, Protein Eng. 7 (1994) 1059–1068.

[13] R.H. Lathrop, T.F. Smith, Global optimum protein threading with gapped alignment and empirical pair score functions, J. Mol. Biol. 255 (1996) 641–665.

[14] L. Lovász, Combinatorial Problems and Exercises, Akadémiai Kiadó, Budapest, 1979.

[15] D. Maier, The complexity of some problems on subsequences and supersequences, J. ACM 25 (1978) 322–336.

[16] T. Nishizeki, N. Chiba, Planar Graphs: Theory and Algorithms, Elsevier, Amsterdam, 1988.

[17] C.H. Papadimitriou, M.Yannakakis, Optimization, approximation, and complexity classes, J. Comput. System Sci. 43 (1991) 425–440.

[18] Y. Xu, E.C. Uberbacher, A polynomial-time algorithm for a class of protein threading problems, Comput. Appl. Biosci. 12 (1996) 511–517.